Character Level LSTM for Shakespearean Text Generation with Neural Sequence Modeling

Hikaru Isayama
University of California San Diego
hisayama@ucsd.edu

Avi Mehta
University of California San Diego
avmehta@ucsd.edu

Wilson Liao
University of California San Diego
wil011@ucsd.edu

Julia Wong
University of California San Diego
juw024@ucsd.edu

Abstract

Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks have demonstrated remarkable capabilities in sequential data modeling, particularly in natural language processing tasks. In this study, we explore character-level text generation in the style of William Shakespeare using both vanilla RNNs and LSTMs, training these models on the TinyShakespeare dataset under different configurations, including teacher forcing and non-teacher forcing. Our experiments evaluate the impact of sequence length, hidden layer size, and temperature scaling on text generation quality. Results show that LSTMs outperform RNNs in capturing long-term dependencies, achieving a peak validation cross-entropy loss of 1.37 compared to 1.53 for the RNN after 10 epochs. When training without teacher forcing, model performance degrades, with loss increasing to 3.30, highlighting the challenges of learning without guided supervision. Text samples generated at different temperature settings demonstrate that lower temperatures produce more structured text, while higher temperatures introduce greater randomness and creativity. These findings emphasize the importance of architectural choices and training strategies in neural text generation, offering insights into balancing coherence and variability in sequence modeling tasks.

1 Introduction

Neural language modeling has significantly advanced with the development of recurrent neural networks (RNNs) and Long Short-Term Memory (LSTM) networks, which excel at capturing temporal dependencies in sequential data. Character-level language models, in particular, offer fine-grained control over text generation and enable networks to learn linguistic patterns at a fundamental level. This study explores the task of generating Shakespearean-style text using character-level RNNs and LSTMs, demonstrating the effectiveness of deep learning models in stylistic text synthesis.

Generating realistic and coherent text requires capturing long-term dependencies within sequences, a task that traditional RNNs struggle with due to the vanishing gradient problem. LSTMs mitigate this issue by incorporating gating mechanisms that regulate information flow across time steps. In this work, we implement and compare character-level RNN and LSTM models trained on the TinyShakespeare dataset, evaluating their ability to learn and generate Shakespearean text. We further investigate the impact of training methodologies, including teacher forcing and free-running inference, to assess the stability and quality of generated text.

Our study aims to answer several key questions: (1) How does the choice of model architecture (RNN vs. LSTM) affect the quality of text generation? (2) How do training strategies, particularly teacher forcing, influence learning dynamics? (3) How does adjusting the temperature parameter

Submitted to the Workshop on Instruction Tuning and Instruction Following at NeurIPS 2023. Do not distribute.

in probabilistic sampling affect the creativity and coherence of generated text? By conducting systematic experiments across different model configurations, we provide insights into best practices for character-level language modeling and highlight trade-offs between deterministic and stochastic text generation.

2 Related Work

Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks have been widely used for sequential data modeling, particularly in natural language processing (NLP) tasks such as text generation. Early studies by Elman (1) demonstrated the capability of simple recurrent networks to capture temporal dependencies, laying the foundation for modern sequence modeling techniques. However, traditional RNNs suffer from the vanishing gradient problem, which limits their ability to learn long-range dependencies (2).

To address this issue, Hochreiter and Schmidhuber (3) introduced LSTMs, which incorporate gating mechanisms to regulate information flow and prevent gradient degradation over long sequences. Their work has since been instrumental in various text generation applications, including character-level language modeling. By leveraging LSTM architectures, researchers have successfully generated structured text in the style of historical authors, such as Shakespeare, demonstrating the model's ability to capture linguistic patterns and stylistic elements.

Our work builds upon these foundational studies by implementing and comparing character-level RNN and LSTM architectures for Shakespearean text generation. We analyze the impact of training strategies, particularly teacher forcing and free-running inference, and explore the effects of hyperparameter tuning on text quality. This study contributes to the ongoing research in neural text generation by evaluating the trade-offs between coherence and variability in sequence modeling.

3 Methods

In this section, we describe the implementation details of our character-level text generation models using Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. We first outline the training process using teacher forcing, followed by an alternative training approach without teacher forcing.

In Table 1, we have listed the hyperparameters chosen for our initial experiments.

Table 1: Model Hyperparameters

Hyperparameter	**Value**
Sequence Length	16
Batch Size	64
Embedding Dimension	25
Hidden Dimension	150
Number of Layers	1
Learning Rate	0.001

3.1 Training network using Teacher forcing

In the teacher forcing approach, the model is trained by providing the ground-truth character as input at each time step, rather than relying on its own generated predictions. This method accelerates convergence and stabilizes training by reducing error accumulation over long sequences. During training, the model learns to predict the next character given a fixed-length input sequence. We use a cross-entropy loss function and optimize using the Adam optimizer with a learning rate of 0.001. The training dataset consists of character sequences extracted from the TinyShakespeare dataset, where we experiment with sequence lengths of 16, 128, and 512. We track both training and validation loss across multiple epochs and evaluate how different sequence lengths impact model performance.

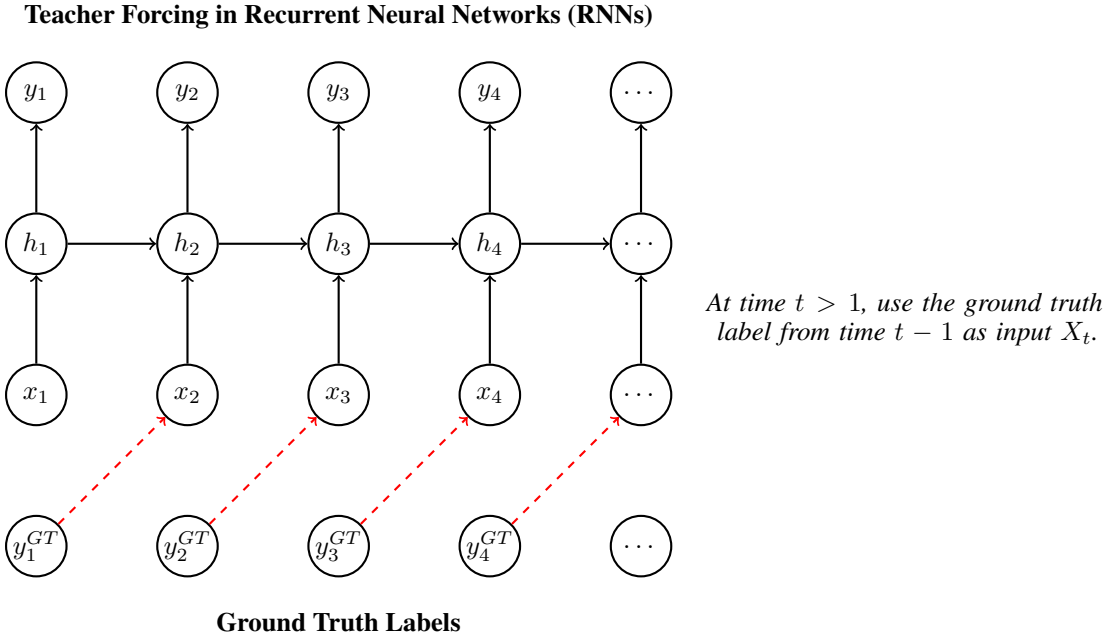


Figure 1: Illustration of teacher forcing in an RNN. The dashed red arrows show where the ground truth labels are used as inputs in the next time step.

3.2 Training network without teacher forcing

To examine the impact of exposure bias, we train a separate set of models without teacher forcing. In this approach, the model generates its own predictions at each time step, using the previous output as input for the next step. This technique better simulates real-world inference conditions but is more challenging to train due to error accumulation. Instead of using the ground-truth character, the model selects the most probable character from the softmax output at each time step. We evaluate how this training paradigm affects convergence, loss stability, and the overall quality of generated text. Given the increased likelihood of error propagation, we experiment with different sequence lengths and introduce gradient clipping to mitigate unstable gradients.

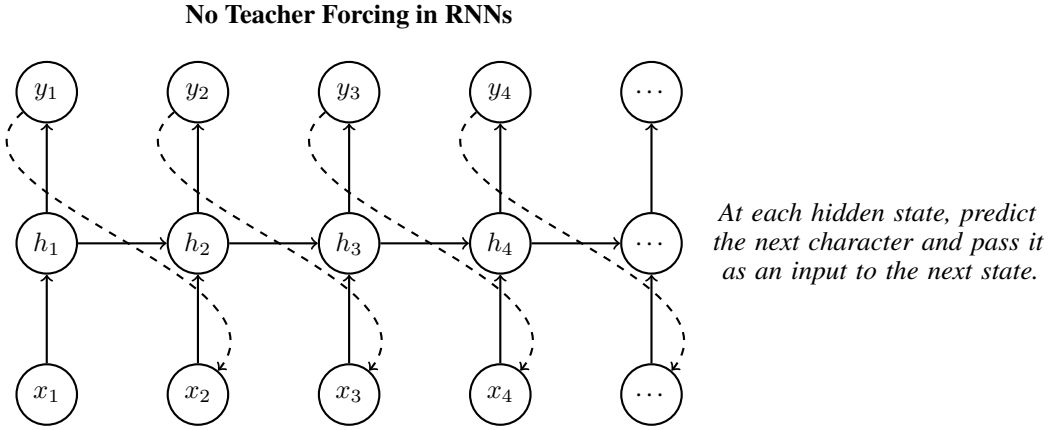


Figure 2: Illustration of RNN training without teacher forcing. Dashed lines indicate that the model's own predictions are fed back as inputs at each timestep.

In Table 2, we have listed the new hyperparameters chosen for our no teacher forcing experiments with sequence length. Since the model was training too slowly using the initial hyperparameters as the model now relies solely on its own predictions instead of ground truth values, we decided to shrinktyt the model complexity by reducing the hidden dimension.

Table 2: No Teacher Forcing Model Hyperparameters

Hyperparameter	Value
Sequence Length	16
Batch Size	64
Embedding Dimension	25
Hidden Dimension	25*
Number of Layers	1
Learning Rate	0.001

3.3 Text Generation

Once training is complete, we use the trained models to generate text in the style of Shakespeare. The network is primed with an initial character sequence, and subsequent characters are generated iteratively. We explore different sampling techniques, including greedy sampling (selecting the highest probability character) and stochastic sampling with temperature scaling. The temperature parameter T controls the randomness of predictions, with lower values producing more deterministic outputs and higher values increasing diversity. We generate text samples for $T = 0.5$, $T = 1.0$, and $T = 2.0$, analyzing how different values affect coherence and creativity.

3.4 Number of Layers (Grad Part)

Finally, we will experiment with increasing the number of layers in the LSTM model to 3 and 4 layers. Our objective is to evaluate how deeper architectures impact the model's ability to capture complex patterns and dependencies in the data. We will also assess the trade-off between improved performance and computational cost to determine the feasibility of using additional layers. This analysis will provide insights into the potential benefits of deeper LSTM architectures for optimizing text generation.

4 Results

In this section, we present the results of our experiments, including baseline model performance, text generation with different temperature settings, and loss comparisons across various training strategies.

Table 3: Baseline Model Performance Metrics

Metric	LSTM	RNN
Validation Loss	1.4104783707354442	1.5485195688864861
Test Loss	1.4076826023221085	1.5416181750565647

To evaluate the impact of increasing the number of neurons in the LSTM, we compared the performance of the model with a hidden size of 170 to our baseline model. As shown in Figure 9, the LSTM with more neurons achieved a validation loss of 1.3745 and a test loss of 1.3782. In contrast, our baseline model, which had fewer hidden units, resulted in a higher validation loss of 1.4105 and a test loss of 1.4077.

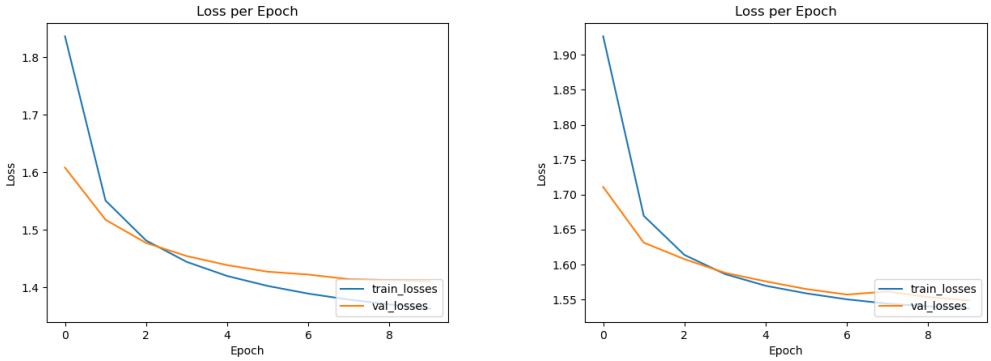
These results indicate that increasing the hidden size improved the model's ability to minimize loss on both validation and test datasets. The lower loss values suggest that the larger model was able to capture more complex patterns in the data, leading to better generalization. However, the improvement is relatively modest, implying that while adding more neurons increased capacity, it did

not drastically enhance performance. This could suggest diminishing returns beyond a certain model size or potential limitations in the dataset that prevent further gains.

Additionally, while the larger model performed better than the baseline, it also required more computational resources and training time. This trade-off between performance gain and efficiency should be considered when selecting model hyperparameters for deployment.

4.1 Baseline Model Performance

Our baseline models consist of a single-layer LSTM and RNN with 150 hidden neurons. After training for 10 epochs on the TinyShakespeare dataset, the performance metrics of this model is shown in Table 3 and Figure 3.



(a) Baseline training and validation losses over epochs for LSTM model

(b) Baseline training and validation losses over epochs for RNN model

Figure 3: Baseline training and validation losses over epochs for LSTM and RNN model with the hyperparameters in Table 1. We will also experiment the results with sequence length of 128 and 512 in the later sections.

4.2 Generated Text Samples

To evaluate the impact of temperature on text generation, we generated three samples using different temperature settings: $T = 0.5$, $T = 1.0$, and $T = 2.0$. In Figure 4, we have a side-by-side comparison between the worst and the best-performing generated text, which we can clearly see how the temperature setting $T = 1.0$ seems to perform much better than when $T = 2$. In addition, text generated with $T = 0.5$ is provided in the last section.

4.3 Loss Comparisons

In this section, we analyze the training loss curves under different configurations to understand how training strategies and architectural choices affect model performance.

4.3.1 No Teacher Forcing

We analyze the impact of training without teacher forcing on model performance. Table 5 summarizes the validation and test loss for the LSTM model trained without teacher forcing, showing a higher loss compared to teacher forcing approaches shown in Figure 3. The increased loss suggests that error accumulation negatively affects long-term dependencies, making convergence slower and less stable.

Figure 5 visualizes the training and validation loss curves, demonstrating that the model struggles to reach lower loss values. This further reinforces the challenges of training without teacher forcing, where the model must rely solely on its previous predictions, leading to compounding errors over time.

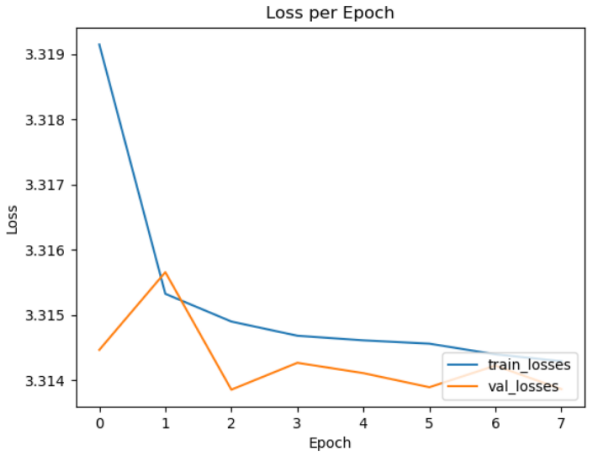


Figure 5: Baseline training and validation losses over epochs for LSTM with no teacher forcing.

4.3.2 Effect of Sequence Length

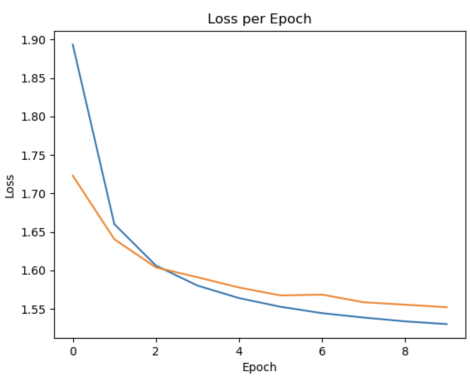
The choice of sequence length plays a significant role in model performance and computational efficiency. Longer sequences allow the model to capture more long-term dependencies, leading to better generalization. However, increasing the sequence length also requires greater computational resources and may lead to diminishing returns if the model struggles to process lengthy contexts effectively.

Table 4 summarizes the validation and test loss for different sequence lengths in both RNN and LSTM models. The results indicate that while longer sequences generally lead to improved performance, the gains become less pronounced at higher lengths, particularly for RNNs. The LSTM model benefits more from longer sequences, achieving a lower final loss compared to the RNN.

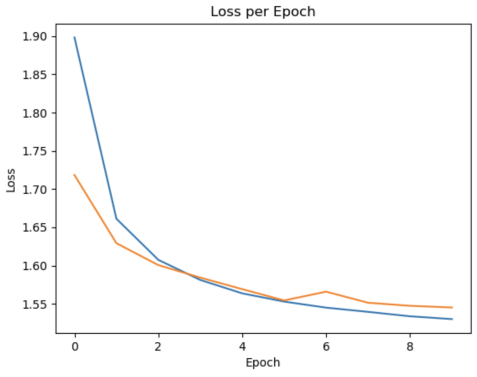
Table 4: Sequence Length Performance Metrics

Metric	Validation Loss	Test Loss
RNN sequence length 16	1.5485195688864861	1.5416181750565647
RNN sequence length 128	1.5520921166915698	1.5551690116943604
RNN sequence length 512	1.5452258569775439	1.5339859970431182
LSTM sequence length 16	1.4104783707354442	1.4076826023221085
LSTM sequence length 128	1.4003919703961787	1.4032720504173655
LSTM sequence length 512	1.3869892421737742	1.3783465808388904

Figures 6 and 7 illustrate the training and validation loss curves for the RNN and LSTM models using sequence lengths of 128 and 512.

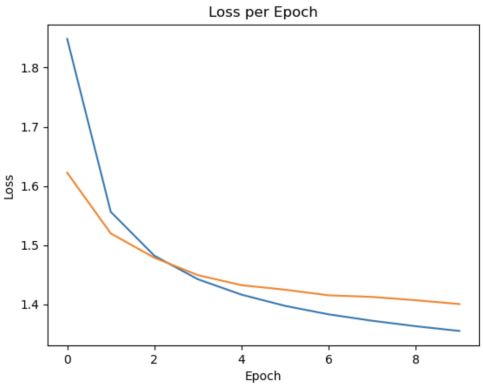


(a) RNN model training with sequence length 128

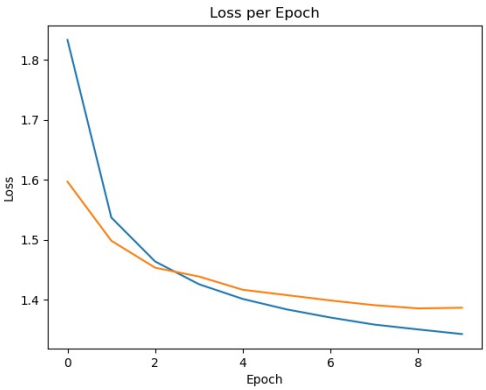


(b) RNN model training with sequence length 512

Figure 6: Training and validation loss over epochs for RNN model using sequence length 128 and 512. The remaining hyperparameters remain the same as shown in Table 1.



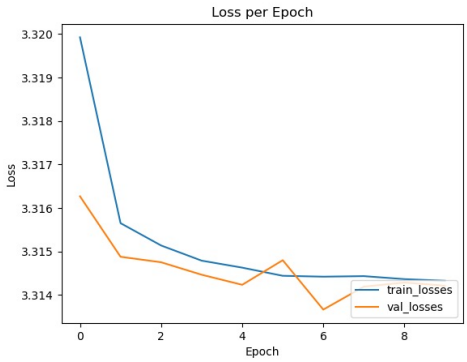
(a) LSTM model training with sequence length 128



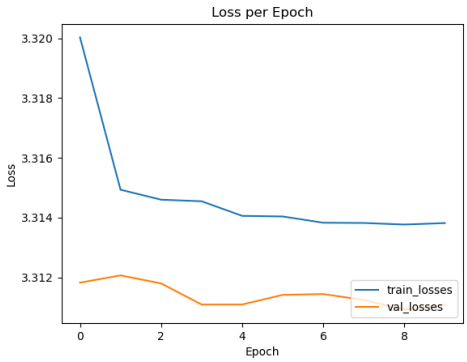
(b) LSTM model training with sequence length 512

Figure 7: Training and validation loss over epochs for LSTM model using sequence length 128 and 512. The remaining hyperparameters remain the same as shown in Table 1.

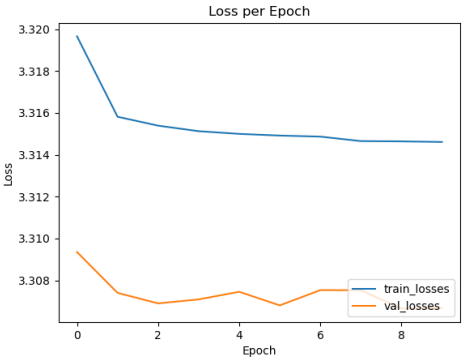
Figure 8 illustrates the training and validation loss curves for the LSTM model without teacher forcing using sequence length 16, 32 and 64, with the alternate hyperparameters set in 2.



(a) LSTM model with no teacher forcing training with sequence length 16



(b) LSTM model with no teacher forcing training with sequence length 32



(c) LSTM model with no teacher forcing training with sequence length 64

Figure 8: Training and validation loss over epochs for LSTM model using sequence length 32 and 64. The remaining hyperparameters as shown in Table 2.

Table 5: Performance metrics for LSTM using no teacher forcing. The hyperparameters for this experiment is shown in Table 2.

Metric (no teacher forcing)	Validation Loss	Test Loss
LSTM sequence length 16	3.3142150513960313	3.3100922104960806
LSTM sequence length 32	3.311085271616469	3.3153672576702946
LSTM sequence length 64	3.306653652393606	3.314864945015002

To further analyze the impact of model capacity on performance, we experimented with an LSTM architecture using an increased number of hidden neurons. Specifically, we set the hidden size to 170, as shown in Figure 9. The motivation behind this experiment was to determine whether a larger model could capture more complex patterns and improve predictive performance.

By increasing the number of hidden units, the model has a higher capacity to learn intricate dependencies in the data. However, this also leads to increased computational cost and the risk of overfitting if the model becomes too complex for the given dataset. As seen in Figure 9, we observed the effect of this change on training and validation loss over multiple epochs, allowing us to assess whether the additional neurons contributed to better generalization or simply fit the training data more closely.

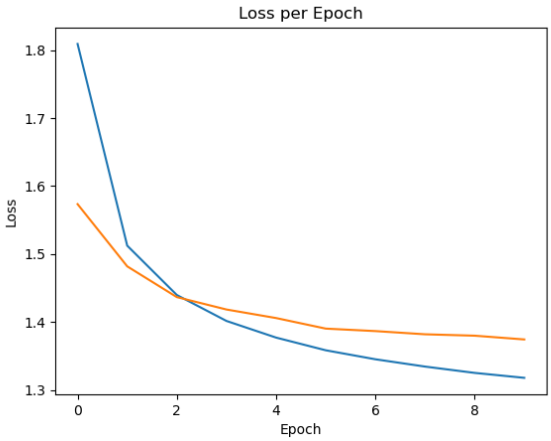


Figure 9: Training and validation losses over epochs for LSTM more neurons (hidden dimension: 170). Achieved a validation loss of 1.3744927933006812 and test loss of 1.3782344693024133

To evaluate the impact of increasing the number of neurons in the LSTM, we compared the performance of the model with a hidden size of 170 to our baseline model. As shown in Figure 9, the LSTM with more neurons achieved a validation loss of 1.3745 and a test loss of 1.3782. In contrast, our baseline model, which had fewer hidden units, resulted in a higher validation loss of 1.4105 and a test loss of 1.4077.

These results indicate that increasing the hidden size improved the model's ability to minimize loss on both validation and test datasets. The lower loss values suggest that the larger model was able to capture more complex patterns in the data, leading to better generalization. However, the improvement is relatively modest, implying that while adding more neurons increased capacity, it did not drastically enhance performance. This could suggest diminishing returns beyond a certain model size or potential limitations in the dataset that prevent further gains.

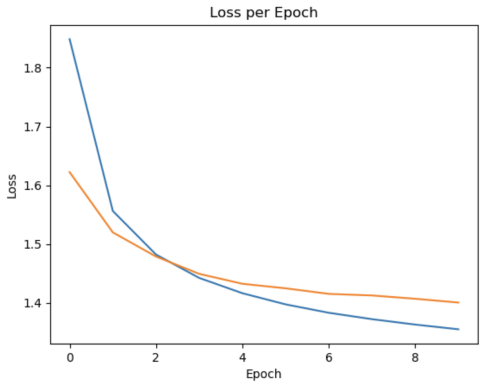
Additionally, while the larger model performed better than the baseline, it also required more computational resources and training time. This trade-off between performance gain and efficiency should be considered when selecting model hyperparameters for deployment.

4.3.3 Effect of Number of Layers (Grad Part)

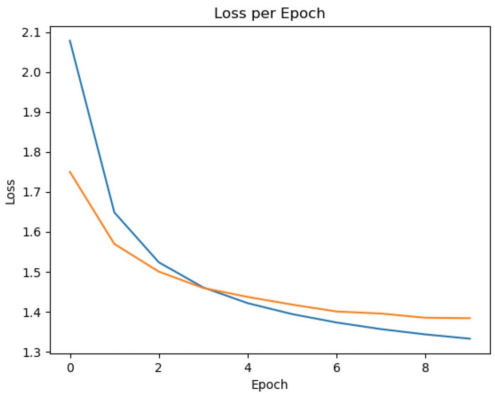
In this section, we experimented with increasing the number of layers in the LSTM model to 3 and 4 layers. Our observations revealed that adding more layers significantly enhanced the model's performance, likely due to its increased capacity to capture complex patterns and dependencies in the data. While this improvement in performance was substantial, the associated increase in runtime was relatively modest, suggesting that the trade-off between performance and computational cost was favorable. This finding highlights the potential of deeper architectures for further optimizing the model's effectiveness.

Table 6: Num Layers Performance Metrics

Metric	LSTM 3 layers	LSTM 4 layers
Validation Loss	1.364118563875821	1.3840815685839059
Test Loss	1.357047081267225	1.3828500149855993



(a) Training and validation losses over epochs for LSTM model with 3 layers



(b) Training and validation losses over epochs for LSTM model with 4 layers

5 Discussion

The performance of the LSTM model was notably better than that of the RNN model, showcasing its capability to handle sequential data more effectively. An increase in the sequence length generally led to improved performance across both models, as the additional contextual information helped the models better capture dependencies in the data. However, it is important to note that the performance gains achieved with longer sequences were relatively modest when compared to the significant increase in the computational resources and runtime required by the model. This trade-off highlights the need to balance sequence length with computational efficiency when optimizing model performance.

As seen in Table 4, as we experiment with longer sequence lengths, the loss of both RNN and LSTM models show no significant improvement. In theory, longer sequence lengths during training allows the LSTM model to learn long term dependencies. With a longer sequence length, back propogation through time (BPTT) is able to update weights of hidden states in earlier positions to "link" two related tokens that are far apart. However, increase in sequence length can still put a strain on the gradient and cause vanishing gradients, and the result of the training is heavily dependent on the frequency of long term dependencies in the training data. In our case, there are some reasons why increased sequence length did not benefit our model.

The first possible reason is that the data does not contain as much long term dependencies, thus, there is not much more relevant context for the model to capture in a longer context window. English contains some degree of structural long term dependencies between characters (ex. opening and closing quotation marks), but little semantic long-term dependencies between characters as a word is the smallest unit capable of holding meaning. As such, increase in sequence length to a certain degree may help generate text that structurally resembles English, but it will not make the model generate more sensible text that will reduce loss.

Another possible reason is that the model has hit a bottleneck with its current size, and in order to achieve a lower loss, we must prioritize expanding other size-related hyperparameters. We can see this in the effect of implementing an LSTM with 3 and 4 layers in Table 6: both validation and test loss went down from the one layer model to multi-layered model. Along with increased number of parameters, the model should be able to capture more accurate representations of the training text that's better than those captured with only increased sequence length.

Lastly, it is also possible that the training dataset is large enough that the model generalizes well even with a short sequence length. As discussed earlier, in character-level models, primary structure of the English language such as word formation (few characters), punctuation(few characters), and common phrase patterns(2-3 words) might be learned effectively even with shorter sequences.

The performance of a smaller LSTM model with various sequence length showed the same learning results(Table 5) as discussed above, with miniscule improvement across loss values when looking solely at the values. However, the three graphs show a notable decrease in overfitting with increased

sequence length: the longer the context window, the lower validation loss is relative to training loss. This confirms the theory, perhaps better than our experiment of sequence length with teacher forcing, that a longer sequence length helps the model generalize better.

Among the four temperature settings (T = 2, T = 1, T = 0.5, and the real text), the T = 1 output is the best-performing in terms of coherence, grammatical structure, and overall readability. While the text still contains some hallucinations and inconsistencies, it preserves the flow of Shakespearean dialogue much better than T = 2 and introduces more variability than T = 0.5, making it more natural and dynamic.

The performance of the generated text varies significantly based on the chosen temperature setting. $T = 2$ is the worst-performing, as it produces highly incoherent and nonsensical text filled with a random mix of letters, symbols, and broken words. The high temperature **introduces excessive randomness**, causing the model to lose control over sentence formation, making the output look like gibberish rather than structured language. An example of the output at $T = 2$ is:

```
QEoBclwer:, 't've! Plinve, Thebp he Rexetant kilden hEng.-.'!
```

In contrast, $T = 1$ is the best-performing setting because it achieves a balance between randomness and coherence, allowing for **creative yet structured text generation. The output at this temperature follows a Shakespearean style, with readable dialogue and logical progression. Although the sentence structure is somewhat off, it **resembles Shakespearean writing and maintains readability. An example of the output at $T = 1$ is:

```
O say the blust to her, I say the better sight of him to the come and a
shown and spries to your say!
```

On the other hand, $T = 0.5$ produces more structured and grammatically correct text but is overly conservative, resulting in repetitive phrasing and a lack of creativity. At this setting, the model is less likely to explore diverse word choices, leading to text that, while readable, is rigid and formulaic. An example of the output at $T = 0.5$ is:

```
Who shall and hath of the sat That the battle state beat that
the dost that the well the seem and so more to the strength a thouse.
```

The results demonstrate that temperature selection is crucial in text generation, directly impacting the balance between coherence and diversity. Lower temperatures ($T = 0.5$) produce structured but rigid text, while higher temperatures ($T = 2$) introduce excessive randomness, resulting in nonsensical output. The optimal temperature ($T = 1$) strikes the best balance, generating text that is both structured and creatively varied. While the generated text still lacks the depth and fluency of real Shakespearean writing, fine-tuning temperature settings allows for a significant improvement in quality. Future enhancements could explore additional model refinements, such as context-aware training and specialized dataset tuning, to further bridge the gap between AI-generated and human-written text.

6 Contributions

Wilson: Debugged models, the train file, experimented with different sequence lengths, and number of layers for the grad portion. I affirm all group members contributed equally.

Julia: Wrote the first draft of the models and train.py, as well as the general set-up of the PA. Generated text in section 4.2. Helped format the write-up and wrote parts of the discussion section. I affirm all group members contributed equally.

Sean: Worked on no teacher forcing section and collaborated with team to work on train.py and main.py files. Contributed to all sections of the write up. I affirm all group members contributed equally.

Avi: Worked on debugging the model architecture and train.py file. Worked on creating evaluate.py file which generated text. I affirm all group members contributed equally

References

[1] Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science*, 14(2), 179–211.

[2] Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (1998). Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies. *A Field Guide to Dynamical Recurrent Neural Networks*, 237–243.

[3] Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780.

7 Generated Test

Along with the two generated text provided in Figure 4, there is one more generated text with temperature 0.5.

```
Macbeth
 by William Shakespeare
 Edited by Barbara A. Mowat and Paul Werstinesce this woil of the beat of your sooth
       of her may shall be so do the pooling of the sweet commorted them with the may
       be be that a shall be so mistance; with the soldier to storm, and we so see to
       may that I have mean of the set the heart,
And in so had are the such bir to the say that is so not to must to the prove to the
      common to be so know the hand me to to good cousin last the still to the lies.

FRIAR LAURENCE:
If you to a slood on the nothing boot.

GRUMIO:
Who shall and hath of the sat
That the battle state beat that the dost that the well the seem and so more to the
    strength a thouse.

PARIS:
If in the world, and the death o'er,
And do rest thou to the wonder and so even the soul.

PERDITA:
O say the blust to her, I say the better sight of him to the come and a shown and
    spries to your say!

QUEEN ELIZABETH:
Come, where was the sir; it you may the ears to come to the beat of the bound the
    poor our sun a shall be so see the rest,
And my son or the sin.

Clown:
What now you are
```

Figure 4: Comparison between worst and best-performing generated text ($T = 2$, $T = 1$).

Worst-Performing Generated Text ($T = 2$)

```
Macbeth
 by William Shakespeare
 Edited by Barbara A. Mowat and Paul
    Werstine.

QEoBclwer:
,'t've! Plinve,
Thebp he Rexetant kilden hEng.-.'!
I weU
He best owh, remussovechible say;: yo.
Slah, closal expoul: I, oncemen,-- Ke,
    ark?'
Quaf.
Wo?'dea,
as formiy!
Go.'y.
fink wast Tu-ded Alpesy!
To st'den fea om. The launcape plusk,
    till, ' mixtels.
I vowpled had fewot
stak
his't  lonk
Avingsed,rfoks,
Rae atyve, therrday; courqueverr-
    pEstaps: a!' tretend,ANol, bubh
    Jupiyarloe
Bole, we'd and Ria, soxdied:'.
'God a mwavy, ender Graslazy!:--whymeo
    ?!
Battle! Ewwyan! O, ip he ofny foentaih
her skaky?

 RIw CY:BTifrienoy, We'gpuoots!
I? oy, FsHarf
who Nay me jalifatys: mre me my sWfemk,
     thrich taxhossrasy;us you;
Indeeciuse, on muser ;omby: bres, my:
PlyitiI toa oondirinn
MrorisanpRomience;
Whowefars.
I'm, gracuber;
Dle's to cape,
Horisedshiaf tUomin h shadop
Fengry. O 'fous'fe; Has I puspyqt:
lad!
Wlonc let se Bitre clat?
'Overois:
To, Edwqizes'd't! You, a jolb;
Hees:
It a
valur's flikio
Yegoois' brerews.

Fintioot: I, any, 'taby Wter.

JrdcI. Grits tash, V om:
So adwahl.
```

Best-Performing Generated Text ($T = 1$)

```
Macbeth
 by William Shakespeare
 Edited by Barbara A. Mowat and Paul
    Werstine. The dost, gentle a longey
By those no hour lish and thereen
By is lillow I streck thy brother from
    hatad.

CYBARILA:
My tesh, it, as day the depusio
Hewh thin your hand't of God the
    cramous
Duke then or him.

AUFINIUS:
Go, I, that so will not pray your town
    me of a would,
Pread thou with the ears,
Sweice say's bill and mothmen:
Ny unlair me
One or she thou wother to your camns,
    stisting sweet;
Orance hear I do hort--'s brad; as the
    wepchio', what we be:
Take his lovement:
Or should ere we'll here own her treat'
    d, with rast,
Nor hast so? bethask'd dutes of may
    common of arm that,
At well and sever, that hear voicion
With as other light, you soldier, I may
     marter?
Nock moy doth this'st ound to thy sweet
     and heart
Hath thou art.

CHARTIAN:
I show'st your was
Scenger'st; 'tis the sleat, but
    idderable our sold
To bat nobla destry.

FRIZH ED XINCENTIO:
And to so; I ald is course: and other
    cannot, and seef thou?

CLYSRES:
What distagry wear the sir,
Did frine it bear esber of tell the
    coun
```